Implantation de la couche algébrique à base de règles de réécriture dans le nouvel APLUSIX

Denis Bouhineau

Implantation de la couche algébrique à base de règles de réécritures dans le nouvel APLUSIX.

Denis Bouhineau
IRIN - Université de Nantes
2, rue de la Houssinière, BP 92208, 44322 Nantes cedex 3
tél : 02 51 12 58 40, fax : 02 51 12 58 12,
mél : Denis.Bouhineau@irin.univ-nantes.fr
www.sciences.univ-nantes.fr/info/recherche/ia/projets/aplusix
10 Février 2000

Résumé : Le projet APLUSIX a entrepris d'effectuer un certain nombre d'évolutions ces dernières années pour atteindre les caractéristiques d'un produit visant l'enseignement de l'algèbre utilisable tout au long du cursus scolaire de l'élève au lycée et au collège. Cet article porte sur les modifications et les choix intervenants au niveau symbolique le plus bas concernant les mathématiques dans le logiciel, i.e. au premier niveau « algébrique » du système servant à définir les traitements des expressions manipulées dans le système. A ce niveau, une base de règles de réécritures est utilisée pour décrire les manipulations mathématiques possibles, il lui est associé un mécanisme d'appariement riche et paramétrable intégrant des connaissances mathématiques.

Présentation générale

Depuis 1988 le projet APLUSIX a pour objectif la modélisation des connaissances humaines dans le domaine des mathématiques, et plus précisément de l'algèbre, en vue de la réalisation d'environnements d'apprentissage (ou EIAH) utilisés en classe*. Ainsi et jusqu'en 1998, plusieurs prototypes ont été développés sur Macintosh en Lisp et régulièrement expérimentés par des professeurs de mathématiques (lycée et collège) ainsi que par des psychologues ; depuis, un travail de réalisation est effectué pour atteindre les caractéristiques d'un produit pour remplacer ces prototypes. Les résultats obtenus avec les prototypes sont positifs sous plusieurs aspects : les élèves apprécient le système, les tests et l'analyse effectuées après expérimentations d'après fichiers montrent des progrès, les enseignants notent que le système apporte une structuration du domaine chez les élèves cf. [Nguyen-Xuan et al.-1999].

Les prototypes et le futur produit fonctionnent selon deux modes d'interactions distincts. Le premier mode est un mode d'observation où l'élève est censé regarder le système en train de résoudre un problème et où éventuellement il peut demander des explications. La résolution s'effectue en pas à pas, en utilisant un moteur de résolution pédagogique basé sur un système expert utilisant les connaissances mathématiques acquises au niveau de l'élève, des connaissances stratégiques simples et un contrôleur pédagogique pour vérifier la pertinence des actions effectuées. Le second mode d'interaction est un mode d'action où l'élève est confronté à un problème mathématique avec pour objectif de le résoudre pas à pas. Les pas de l'élève sont à choisir parmi une liste de manipulations algébriques usuelles (utilisation d'une identité remarquable, développement d'un produit de facteur, réduction arithmétique, etc.). Le calcul explicite au niveau arithmétique des produits, sommes, … est effectué par le système garantissant ainsi que les expressions manipulées sont bien toutes équivalentes du point de vue algébrique et interdisant l'introduction d'erreur de 'calcul'. Une analyse des pas de calcul demandés par l'élève est effectuée par le système et éventuellement ceux-ci lui sont refusés s'ils sont erronés du point de vue mathématique ou maladroit du point de vue stratégique. Dans ce mode, l'élève a la possibilité de demander de l'aide. Cette aide est construite par le moteur expert en prenant en compte la résolution entreprise par l'élève.

Pour arriver à obtenir quelques résultats, les prototypes du projet comportent quelques limitations. Tout

* le lecteur connaissant déjà le projet APLUSIX peut commencer sa lecture à la section suivante.

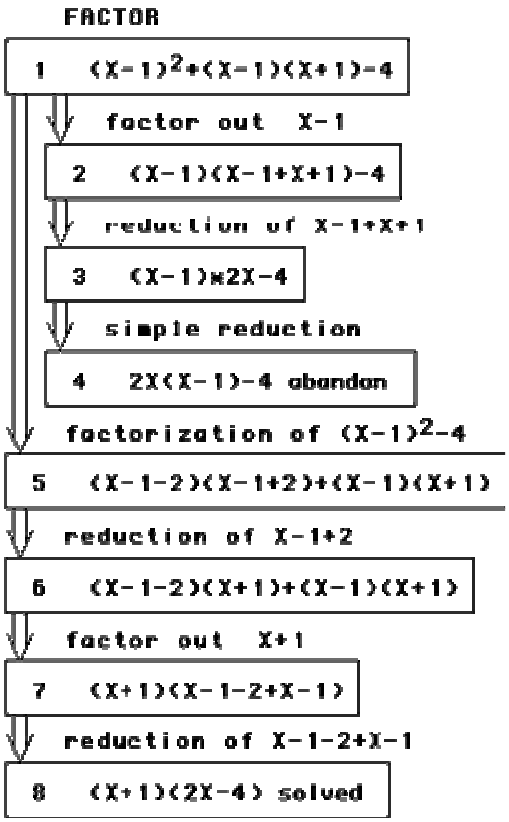

Figure 1 :Utilisation Courante d'APLUSIX

d'abord pour les classes d'âges visées, le niveau scolaire concerné porte essentiellement sur les classes de collège allant de la quatrième jusqu'à la classe de seconde en lycée. En particulier les problèmes et apprentissages concernant les manipulations des fractions rationnelles intervenant en début de collège, et les manipulations algébriques faisant intervenir les discriminants des dernières années au lycée ne sont pas abordées. Ensuite en ce qui concerne la classe de problèmes algébriques traité par APLUSIX l'essentiel du travail porte sur les factorisations de polynômes et les résolutions d'équations polynomiales. Les systèmes d'équations, et les inéquations ne sont pas abordées, par exemple.

Ces limitations énoncées, notons maintenant les points positifs qui ont fait l'objet de travaux particuliers dans le projet APLUSIX. Tout d'abord, APLUSIX n'est pas seulement un modèle d'EIAH ou un outil théorique de construction et de modélisation des connaissances humaines en situation d'apprentissage. C'est aussi un logiciel qui fonctionne réellement et qui a été utilisé pendant de longues périodes en face d'élèves. En particulier en ce qui concerne les réalisations dans le domaine de l'enseignement, ces efforts signifient un travail sur l'interface élève-machine et sur l'ergonomie autant en saisie qu'en affichage. En saisie, des expressions en écriture abrégée sont acceptées, ex : '3x+2y=5', des expressions ou sous-expressions avec une syntaxique non valide du point de vue mathématique ne peuvent être sélectionnées, ex : 'x+2' ne peut être sélectionnée comme sous-expression de '3x+2y'. En affichage les expressions sont dessinées graphiquement en deux dimensions. Au niveau stratégique le développement des calculs et des recherches se fait selon un arbre avec possibilité de revenir en arrière, d'avoir plusieurs chemins parcourus en parallèle, etc.

Ensuite, en terme d'EAIH, APLUSIX se trouve à mi-chemin entre les tuteurs et les micro-mondes. Comme tuteur il est capable d'apporter un diagnostic pédagogique vis-à-vis d'une activité mathématique. Il est capable également d'organiser une séquence d'exercices selon une progression s'appuyant sur les compétences de l'utilisateur. Cela permet d'effectuer une conduite de séances pour un professeur. Comme micro-monde, il garde l'ouverture face aux énoncés et aux situations proposées. Les manipulations algébriques permises sont paramétrables, la classe d'énoncés acceptés en entrée est ouverte, les solutions acceptables ne sont pas limitées à une seule forme particulière, elles sont mathématiquement définies et évaluées comme telles, par exemple dans la situation la plus courante, i.e. la factorisation, la solution d'un problème de factorisation est définie comme le produit de polynômes du premier degré mis sous forme réduite.

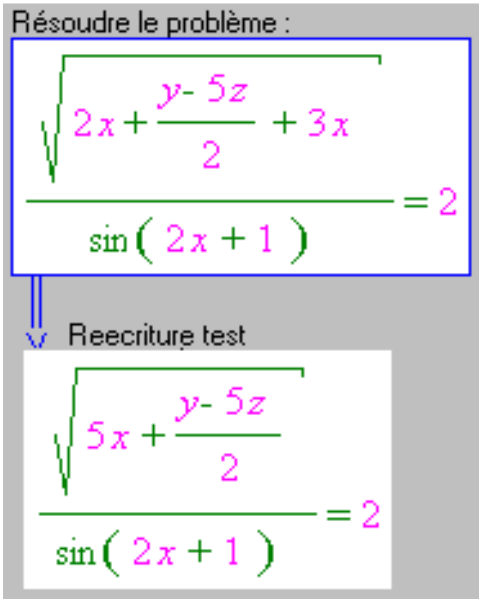

Figure 2 : Nouvelle présentation

Enfin des efforts ont été effectués pour assurer une forme de transparence dans la conception d'APLUSIX, afin que les utilisateurs ou d'éventuels analystes puissent avoir une idée assez précise du comportement. Ainsi un modèle décrit à niveau connaissance du prototype de 1994 a été publié *in extenso* dans [Nicaud-1994]. Ce modèle est disponible sur internet http://www.sciences.univ-nantes.fr/info/recherche/ia/projets/aplusix/modkaplx.htm . Dans le futur produit la base de règles de réécritures utilisée pour décrire le comportement du logiciel est accessible à l'utilisateur et rédigée sous forme lisible. Une modification de cette base de règles - au risque de l'utilisateur- transforme le comportement du logiciel. La syntaxe des expressions mathématiques utilisée est souple et naturelle comme il est exprimée plus haut. Attention, cependant, cette transparence vise un public averti, et les modifications de la base de règle visent un public expert.

Nous pensons qu'ainsi des versions adaptées à des situations particulières peuvent être réalisées de manière indépendante, par exemple une version pour physiciens ou une version pour une branche précise des mathématiques de plus haut niveau.

L'objectif à moyen terme pour le projet APLUSIX est la production d'un système ouvert largement diffusé et capable d'apporter une aide en apprentissage sur un domaine traité pendant plusieurs années, cette aide consistant principalement à encadrer les élèves en train de résoudre des exercices de type algébriques ouverts.

Base de règles de réécritures

Dans l'architecture d'APLUSIX (cf. figure 3) le premier niveau concernant les mathématiques comporte la définition de structures de données pour représenter les expressions arithmétiques courantes (sous forme d'arbres, avec pour feuilles les symboles et constantes numériques, pour nœuds les opérateurs : le moins et les parenthèses étant considérés comme des opérateurs unaires, la barre de fraction et le signe égale étant considérés comme des opérateurs binaires et la multiplication et la somme comme des opérateurs n-aires) et la définition des manipulations algébriques de base possibles. Ces manipulations sont décrites sous forme de règles de réécritures accessibles en clair dans un fichier de configuration d'APLUSIX. On notera deux ruptures pour APLUSIX, d'une part l'utilisation de règles de réécritures au lieu de fonctions internes au code pour décrire les manipulations de base, d'autre part la transparence et l'accessibilité de ces définitions à un niveau de connaissance

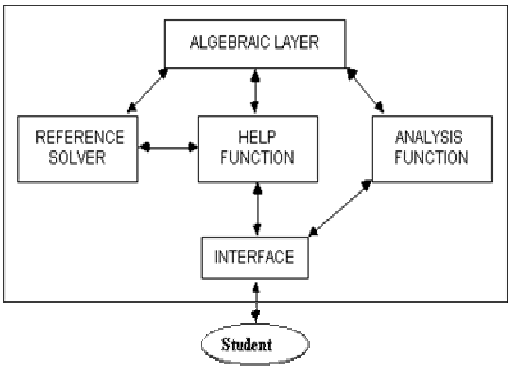

Figure 3 : Architecture d'APLUSIX

```
RègleRéécriture ::=
  Réécriture ConditionFacultative ';'
 ;

Réécriture ::=
  Expression '--->>>' Expression
 ;

ConditionFacultative ::=
   '|' Condition
 |                  // rien
 ;
```

Figure 4 : grammaire des règles de réécriture

moins spécialisé.

Les règles de réécritures sont décrites par la grammaire données à la figure 4. Lorsqu'elles comportent une 'Condition' elles ne sont applicables qu'à partir du moment où cette condition est vérifiée. Les conditions apparaissant dans cette grammaire sont de deux genres différents, les conditions de 'type', et les conditions de 'calcul'. Les conditions de 'type' imposent qu'une variable de réécriture soit d'un type algébrique particulier, comme une somme, une racine carrée, etc. et les conditions de 'calcul' imposent que le résultat d'un calcul numérique soit égal à une certaine expression. Ces conditions de 'calcul' permettent de donner une valeur à de nouvelles variables de réécriture ou de vérifier des contraintes numériques.

Les règles de réécritures sont regroupées en classes, une règle peut apparaître éventuellement dans plusieurs classes. Parmi les classes il y a : les réductions de bases (exemples : 1*a→a, a+b→c quand c est calculable), les factorisations, les développements, les identités remarquables, les productions de carrés, les mises sous forme normale.

Pour chaque classe de règles de réécritures, plusieurs questions formelles se posent. Est-ce que le système de réécriture composé de ces règles est confluent ? Est-ce que l'application de ces règles de réécritures termine ? Vis-à-vis d'un problème donné, l'emploi des règles d'une classe donnée permet-il d'obtenir les solutions existantes (complétude) et les solutions données par le système sont-elles justes (correction) ?

Plus délicat, comment les éléments de la théorie des réécritures permettent-ils de caractériser ou d'établir une théorie sur la résolution des problèmes algébriques dans le contexte où se trouvent les élèves.

Par exemple, pour la classe des réductions, l'ensemble des règles est un système confluent avec terminaison.

Remarque importante : l'ensemble des règles de réécriture définies conserve l'expression mathématique intrinsèque du point de vue algébrique. Ainsi l'utilisation de ces règles permet de travailler par équivalence algébrique. Cette propriété permet d'assurer la correction des calculs pour tous les problèmes que l'on nommera algébrique, c'est-à-dire pour tous les problèmes qui ont un même ensemble de solutions pour des formulations algébriques équivalentes. Exemple : « factoriser » est un problème algébrique. Contre-exemple : « compter le nombre de 'x' dans une équation » n'est pas un problème algébrique. Dans un proche avenir, il ne nous semble pas nécessaire de dépasser le cadre de ces problèmes algébriques.

```
a+0 --->>> a;
--a --->>> a;
a*b+a*c --->>> a(b+c);
(a-b)(a+b) --->>> a^2-b^2;
a+b --->>> c
  | Type a (1)
  | Type b (1)
  | Calcul c (a+b);
(-a)^n --->>> -a^n
  | Type n (1)
  | Calcul 1 (Modulo n 2);
(a)+b --->>> a+b
  | Type a (1+1);
a*b+a*c --->>> a*d
  | Type b (1)
  | Type c (1)
  | Calcul d (b+c);
```

Figure 5 : Exemples de règles de réécritures

En relation directe avec les règles de réécritures présentées ici, la section suivante concernera les mécanismes d'appariement utilisés pour appliquer une règle de réécriture. Au préalable, également en relation directe, mais en deux phrases seulement, notons tout d'abord que des règles de réécritures usuelles, données en intention, par exemple une règle de développement généralisée : `c(a+b+…+z)→c.a+c.b+…+c.z`, ne sont pas formulables dans la grammaire proposée. Pour atteindre ces règles on peut songer à des plans, i.e. : suite de règles terminant ou ne terminant éventuellement pas. Notons également que ces règles sont données à un niveau élémentaire, pour assurer une maîtrise maximale du système, et par conséquent peuvent alourdir les calculs. Un mécanisme de visibilité permet de faire disparaître, selon le niveau de l'élève, dans un ensemble de calculs les pas de calculs ayant moins d'importance et censés être immédiat à un niveau donné.

Mécanisme d'appariement

L'application des règles de réécritures nécessite l'emploi d'un mécanisme d'appariement qui, étant donné une expression candidate C, détermine si une règle R est applicable et donne dans ce cas une liste de substitutions S prenant en compte toutes les variables apparaissant en partie gauche de la règle de réécriture que l'on appellera le modèle de l'expression à apparier. Le mécanisme d'appariement utilisé dans APLUSIX est une forme adaptée aux mathématiques des algorithmes d'unification que l'on trouve dans les langages de programmation logique ou dans les langages de shell ou de manipulation de chaînes de caractères avec jokers et expressions régulières qui sont eux adaptés à l'informatique et aux traitements des langues.

Ainsi l'appariement recherché doit pouvoir éventuellement prendre en compte

Candidat	Modèle	Substitution
3x+5+10x+15	a.c+b.c	a:3,b:10,c:x
16-(x-5)(x-5)	a^2-b^2	a:4,b:(x-5)
1+2+3+4	a+b	a:1, b:2+3+4

Figure 6 : Exemples d'appariement

l'associativité des opérateurs, la commutativité, la présence d'éléments neutre. L'appariement doit pouvoir également s'effectuer sur une sous-expression propre du candidat ou à un facteur près, i.e. : une constante multiplicative positive ou un signe. Enfin, certains règles de réécritures expriment des manipulations sur des concepts, comme le concept de carré dans la règle $a^2-b^2 \rightarrow (a-b)(a+b)$, et dans ce cas l'appariement doit s'effectuer à un concept près.

L'implantation de chacune des particularités décrites précédemment de l'appariement mathématique présente des difficultés techniques non négligeables et dépend fortement des représentations internes choisies pour les expressions, elle dépend aussi fortement selon que l'on recherche un appariement à partir du candidat ou de la règle de réécriture (le plus efficace semblant d'effectuer une recherche commune). Dans tous les cas, il semble nécessaire, d'une part de pouvoir limiter l'exécution d'un appariement trop riche, d'autre part de connaître les mécanismes exécutés pour arriver à un appariement donné.

Certains choix temporaires ont été faits pour mettre en place un appariement de cet ordre dans APLUSIX. La gestion des éléments neutres et de la commutativité est effectuée par des règles de réécriture ; l'associativité et la recherche de sous-expressions est prise en compte dans le noyau dur de la programmation de l'appariement ; la recherche de concepts (comme le concept de carré) s'effectue en chaînage arrière.

Quant à la recherche d'appariement à un facteur près, c'est de loin le mécanisme le plus délicat à implanter, il s'effectue en

deux passes. La première passe décore l'arbre de l'expression des facteurs possibles apparaissant dans les sous-expressions, puis la seconde passe tente un appariement à un facteur près parmi ces facteurs. Pour limiter la complexité algorithmique, la recherche de facteurs multiplicatifs et l'appariement ne s'effectuent que sur les nombres positifs. L'appariement à un signe près n'est pas pris en compte pour le moment. Un modèle d'appariement à un facteur multiplicatif près quelconque (positif ou négatif) a été étudié et semble fonctionner sur une structure interne différente pour les expressions et sera peut-être mis en place ultérieurement (sans opérateur '-', mais avec des entiers positifs ou négatifs comme coefficients numériques).

Mot de la fin

Le projet APLUSIX pour l'obtention d'un produit diffusable arrive bientôt à son terme. C'est avec impatience que nous attendons de rassembler tous les morceaux pour effectuer les premiers tests en grandeur nature. En particulier, nous allons pouvoir observer comment notre souhait de laisser le plus de choses ouvertes permet d'élargir l'utilisation d'APLUSIX, si ces ouvertures sont suffisantes.

Références

[Boizumault-1988] Boizumault P., PROLOG l'implantation, Masson Ed, p13-27.

[Goldstein-1982] Goldstein I.P., The genetic graph : a representation for the evolution of procedural knowledge. Intelligent Tutoring Systems, Academic Press.

[Nicaud-1994] Nicaud J.F., Modélisation en EIAO, les modèles d'APLUSIX. Revue Recherche en Didactique des Mathématiques, Vol 14, n° 1-2, p 67-112.

[Nicaud et al.-1999] Nicaud, J.F., Bouhineau, D., Varlet, C., Nguyen-Xuan, A., Towards a Product for Teaching Formal Algebra. Proceedings of Artificial Intelligence and Education, Le Mans.

[Nguyen-Xuan et al.-1999] Nguyen-Xuan, A., Bastide, A., Nicaud, J.F., Learning to Solve Polynomial Factorization Problems : by Solving Problems and by Studying Examples of Problem Solving, with an ILE.. Proceedings of Artificial Intelligence and Education, Le Mans.